

Zilog

```

ZZZZZZZ  88888  000
  Z      8      8  0  0
  Z      8      8  0  0  0
  Z      88888  0  0  0
  Z      8      8  0  0  0
  Z      8      8  0  0
ZZZZZZZ  88888  000

```

Z80 MICROPROCESSOR Instruction Set Summary

<-- A11		1		40		A10 -->
<-- A12		2		39		A9 -->
<-- A13		3		38		A8 -->
<-- A14		4		37		A7 -->
<-- A15		5		36		A6 -->
--> CLK		6		35		A5 -->
<--> D4		7		34		A4 -->
<--> D3		8		33		A3 -->
<--> D5		9		32		A2 -->
<--> D6		10	Z80	31		A1 -->
+5V		11		30		A0 -->
<--> D2		12		29		GND
<--> D7		13		28		RFSH -->
<--> D0		14		27		M1 -->
<--> D1		15		26		RESET <--
--> INT		16		25		BUSRQ <--
--> NMI		17		24		WAIT <--
<-- HALT		18		23		BUSAK -->
<-- MREQ		19		22		WR -->
<-- IORQ		20		21		RD -->

Written by	Jonathan Bowen	
	Programming Research Group	
	Oxford University Computing Laboratory	
	8-11 Keble Road	
	Oxford OX1 3QD	
	England	
	Tel +44-865-273840	
Created	August 1981	
Updated	April 1985	
Issue	1.3	Copyright (C) J.P.Bowen 1985

Mnemonic	SZHPNC	Description	Notes
ADC A,s	***V0*	Add with Carry	A=A+s+CY
ADC HL,ss	***V0*	Add with Carry	HL=HL+ss+CY
ADD A,s	***V0*	Add	A=A+s
ADD HL,ss	--?-0*	Add	HL=HL+ss
ADD IX,pp	--?-0*	Add	IX=IX+pp
ADD IY,rr	--?-0*	Add	IY=IY+rr
AND s	***P00	Logical AND	A=A&s
BIT b,m	?*1?0-	Test Bit	m&{2^b}
CALL cc,nn	-----	Conditional Call	If cc CALL
CALL nn	-----	Unconditional Call	-[SP]=PC,PC=nn
CCF	--?-0*	Complement Carry Flag	CY=~CY
CP s	***V1*	Compare	A-s
CPD	****1-	Compare and Decrement	A-[HL],HL=HL-1,BC=BC-1
CPDR	****1-	Compare, Dec., Repeat	CPD till A=[HL]or BC=0
CPI	****1-	Compare and Increment	A-[HL],HL=HL+1,BC=BC-1
CPIR	****1-	Compare, Inc., Repeat	CPI till A=[HL]or BC=0
CPL	--1-1-	Complement	A=~A
DAA	***P-*	Decimal Adjust Acc.	A=BCD format
DEC s	***V1-	Decrement	s=s-1
DEC xx	-----	Decrement	xx=xx-1
DEC ss	-----	Decrement	ss=ss-1
DI	-----	Disable Interrupts	
DJNZ e	-----	Dec., Jump Non-Zero	B=B-1 till B=0
EI	-----	Enable Interrupts	
EX [SP],HL	-----	Exchange	[SP]<->HL
EX [SP],xx	-----	Exchange	[SP]<->xx
EX AF,AF'	-----	Exchange	AF<->AF'
EX DE,HL	-----	Exchange	DE<->HL
EXX	-----	Exchange	qq<->qq' (except AF)
HALT	-----	Halt	
IM n	-----	Interrupt Mode	(n=0,1,2)
IN A,[n]	-----	Input	A=[n]
IN r,[C]	***P0-	Input	r=[C]
INC r	***V0-	Increment	r=r+1
INC [HL]	***V0-	Increment	[HL]=[HL]+1
INC xx	-----	Increment	xx=xx+1
INC [xx+d]	***V0-	Increment	[xx+d]=[xx+d]+1

INC ss	-----	Increment	ss=ss+1
IND	?*??1-	Input and Decrement	[HL]=[C], HL=HL-1, B=B-1
INDR	?1??1-	Input, Dec., Repeat	IND till B=0
INI	?*??1-	Input and Increment	[HL]=[C], HL=HL+1, B=B-1
INIR	?1??1-	Input, Inc., Repeat	INI till B=0
JP [HL]	-----	Unconditional Jump	PC=[HL]
JP [xx]	-----	Unconditional Jump	PC=[xx]
JP nn	-----	Unconditional Jump	PC=nn
JP cc,nn	-----	Conditional Jump	If cc JP
JR e	-----	Unconditional Jump	PC=PC+e
JR cc,e	-----	Conditional Jump	If cc JR(cc=C,NC,NZ,Z)
LD dst,src	-----	Load	dst=src
LD A,i	**0*0-	Load	A=i (i=I,R)
LDD	--0*0-	Load and Decrement	[DE]=[HL], HL=HL-1, #
LDDR	--000-	Load, Dec., Repeat	LDD till BC=0
LDI	--0*0-	Load and Increment	[DE]=[HL], HL=HL+1, #
LDIR	--000-	Load, Inc., Repeat	LDI till BC=0
NEG	***V1*	Negate	A=-A
NOP	-----	No Operation	
OR s	***P00	Logical inclusive OR	A=Avs
OTDR	?1??1-	Output, Dec., Repeat	OUTD till B=0
OTIR	?1??1-	Output, Inc., Repeat	OUTI till B=0
OUT [C],r	-----	Output	[C]=r
OUT [n],A	-----	Output	[n]=A
OUTD	?*??1-	Output and Decrement	[C]=[HL], HL=HL-1, B=B-1
OUTI	?*??1-	Output and Increment	[C]=[HL], HL=HL+1, B=B-1
POP xx	-----	Pop	xx=[SP]+
POP qq	-----	Pop	qq=[SP]+
PUSH xx	-----	Push	-[SP]=xx
PUSH qq	-----	Push	-[SP]=qq
RES b,m	-----	Reset bit	m=m&{~2^b}
RET	-----	Return	PC=[SP]+
RET cc	-----	Conditional Return	If cc RET
RETI	-----	Return from Interrupt	PC=[SP]+
RETN	-----	Return from NMI	PC=[SP]+
RL m	**0P0*	Rotate Left	m={CY,m}<-
RLA	--0-0*	Rotate Left Acc.	A={CY,A}<-
RLC m	**0P0*	Rotate Left Circular	m=m<-
RLCA	--0-0*	Rotate Left Circular	A=A<-

Mnemonic	SZHPNC	Description	Notes
RLD	**0P0-	Rotate Left 4 bits	{A, [HL]}={A, [HL]}<- ##
RR m	**0P0*	Rotate Right	m=->{CY,m}
RRA	--0-0*	Rotate Right Acc.	A=->{CY,A}
RRC m	**0P0*	Rotate Right Circular	m=->m
RRCA	--0-0*	Rotate Right Circular	A=->A
RRD	**0P0-	Rotate Right 4 bits	{A, [HL]}=->{A, [HL]} ##
RST p	-----	Restart	(p=0H, 8H, 10H, ..., 38H)
SBC A,s	***V1*	Subtract with Carry	A=A-s-CY
SBC HL,ss	***V1*	Subtract with Carry	HL=HL-ss-CY
SCF	--0-01	Set Carry Flag	CY=1
SET b,m	-----	Set bit	m=mv{2^b}
SLA m	**0P0*	Shift Left Arithmetic	m=m*2
SRA m	**0P0*	Shift Right Arith.	m=m/2
SRL m	**0P0*	Shift Right Logical	m=->{0,m,CY}
SUB s	***V1*	Subtract	A=A-s
XOR s	***P00	Logical Exclusive OR	A=Axs

F	-*01?	Flag unaffected/affected/reset/set/unknown
S	S	Sign flag (Bit 7)
Z	Z	Zero flag (Bit 6)
HC	H	Half Carry flag (Bit 4)
P/V	P	Parity/Overflow flag (Bit 2, V=overflow)
N	N	Add/Subtract flag (Bit 1)
CY	C	Carry flag (Bit 0)
<hr/>		
n		Immediate addressing
nn		Immediate extended addressing
e		Relative addressing (PC=PC+2+offset)
[nn]		Extended addressing
[xx+d]		Indexed addressing
r		Register addressing
[rr]		Register indirect addressing
		Implied addressing
b		Bit addressing
p		Modified page zero addressing (see RST)
<hr/>		
DEFB n(,...)		Define Byte(s)
DEFB 'str'(,...)		Define Byte ASCII string(s)
DEFS nn		Define Storage Block
DEFW nn(,...)		Define Word(s)
<hr/>		
A B C D E		Registers (8-bit)
AF BC DE HL		Register pairs (16-bit)
F		Flag register (8-bit)
I		Interrupt page address register (8-bit)
IX IY		Index registers (16-bit)
PC		Program Counter register (16-bit)
R		Memory Refresh register
SP		Stack Pointer register (16-bit)
<hr/>		
b		One bit (0 to 7)
cc		Condition (C,M,NC,NZ,P,PE,PO,Z)
d		One-byte expression (-128 to +127)
dst		Destination s, ss, [BC], [DE], [HL], [nn]
e		One-byte expression (-126 to +129)
m		Any register r, [HL] or [xx+d]
n		One-byte expression (0 to 255)
nn		Two-byte expression (0 to 65535)
pp		Register pair BC, DE, IX or SP
qq		Register pair AF, BC, DE or HL
qq'		Alternative register pair AF, BC, DE or HL
r		Register A, B, C, D, E, H or L
rr		Register pair BC, DE, IY or SP
s		Any register r, value n, [HL] or [xx+d]
src		Source s, ss, [BC], [DE], [HL], nn, [nn]
ss		Register pair BC, DE, HL or SP
xx		Index register IX or IY
<hr/>		
+ - * / ^		Add/subtract/multiply/divide/exponent
& ~ v x		Logical AND/NOT/inclusive OR/exclusive OR
<- ->		Rotate left/right
[]		Indirect addressing
[]+ -[]		Indirect addressing auto-increment/decrement
{ }		Combination of operands
#		Also BC=BC-1,DE=DE-1
##		Only lower 4 bits of accumulator A used